Customer Database Management System (CDMS)

What is CDMS?

- The Customer Database Management System (CDMS) is a full-stack application developed using SQL, Python, and Streamlit.
- It is designed to manage customer-related operations such as storing customer information, managing products, tracking orders, handling payments, and addressing customer support tickets.

Why CDMS?

- CDMS helps businesses streamline their operations by centralizing all customer interactions in one platform. This system enhances data accessibility, improves customer service, and ensures organized records.

How does CDMS work?

- CDMS operates through three layers:
    - Frontend (Streamlit): Provides an interactive user interface for accessing and manipulating data.
    - Backend (Python): Handles logic, routes user input, and executes database queries.
    - Database (SQL): Stores data in structured tables such as Customers, Products, Orders, etc.

1. Key Features

- Add, View, and Search Customers
- Manage Products and Stock
- Track and Update Orders
- Monitor Payments and Methods
- Manage Customer Support Tickets

2. Code Implementation

Below are the main code components used in the development of CDMS.

2.1 Create virtual environment

- Go to C drive – New folder (name it my projects)-- New folder(name it EMS). Then copy the folder location.
- Then go to the Scripts folder, go to Python 312, then type cmd in the address bar.
- Then type ***python -m venv <Paste the folder location>*** and hit Enter.
- Then go to the Scripts folder of the EMSand type cmd in the address bar.
- The type activate and the virtual environment will be activated.

2.2 Connect MySQL to python

```
import mysql.connector
mydb = mysql.connector.connect(host='local',user='root', password='12345678',database='cdms')
c = mydb.cursor()
c.execute(<write your sql command>)
```

Note below syntax:

To retrieve data:

```
For row in c:
    print(row)
```

To insert data

```
mydb.commit()
```

Syntax to print any row in CMD screen:

```
import mysql.connector
mydb = mysql.connector.connect(host='localhost',user='root', password='12345678',database='cdms')
c = mydb.cursor()
```

```
c.execute("select * from customers")
for row in c:
    print(row)
```

Syntax to print specific row in CMD screen (example if we want to print first row of the table:

```
import mysql.connector
mydb = mysql.connector.connect(host='localhost',user='root', password='12345678',database='cdms')
c = mydb.cursor()
c.execute("select * from customers")
for row in c:
    print(row[1])
```

When we run the above code, it gives output in the form of **‘tuple’.**

Syntax to enter a values in any row

```
import mysql.connector
import datetime
mydb = mysql.connector.connect(host='localhost',user='root', password='12345678',database='cdms')
c = mydb.cursor()

cid=input('Enter your Customer ID')
fn=input('Enter your first name')
ln=input('Enter your last name')
email=input('Enter your Email address')
ph=input('Enter your contact number')
add=input('Enter your address')
issueid=str(datetime.datetime.now())

c.execute("insert into customers values(%s,%s,%s,%s,%s,%s,%s)",(cid,fn,ln,email,ph, add,issueid))
mydb.commit()
print("Customer profile",cid,"status saved at:", issueid)
```

2.3 SQL Code

Paste the SQL code here:

```
CREATE database cdms;

USE cdms;

CREATE TABLE Customers (
   CustomerID INT PRIMARY KEY,
   FirstName VARCHAR(50) NOT NULL,
   LastName VARCHAR(50) NOT NULL,
   Email VARCHAR(100) UNIQUE NOT NULL,
   Phone VARCHAR(15),
   Address TEXT,
   CreatedAt TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

INSERT INTO Customers (CustomerID, FirstName, LastName, Email, Phone, Address, CreatedAt)
VALUES
(001, 'John', 'Doe', 'john.doe@example.com', '1234567890', '123 Main St', '2024-09-27 18:45:33'),
(002, 'Jane', 'Smith', 'jane.smith@example.com', '0987654321', '456 Elm St', '2024-07-15 21:19:17'),
(003, 'Alice', 'Brown', 'alice.brown@example.com', '1112223333', '789 Oak St', '2024-04-27 08:22:58'),
(004, 'Bob', 'Johnson', 'bob.johnson@example.com', '4445556666', '321 Pine St', '2025-03-07
22:00:10'),
(005, 'Charlie', 'Davis', 'charlie.davis@example.com', '7778889999', '654 Maple St', '2024-06-24
21:44:39');

Drop Table Products;
CREATE TABLE Products (
   ProductID VARCHAR(50) PRIMARY KEY,
   ProductName VARCHAR(100) NOT NULL,
   Description TEXT,
   Price DECIMAL(10,2) NOT NULL,
   Stock INT NOT NULL,
   CreatedAt TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

INSERT INTO Products (ProductID, ProductName, Description, Price, Stock, CreatedAt) VALUES
('E102548', 'Laptop', 'High-end gaming laptop', 1200.00, 10, '2024-09-27 18:45:33'),
('H935477', 'Smartphone', 'Latest model smartphone', 800.00, 25,'2024-04-27 08:22:58'),
('C368412', 'Headphones', 'Noise-canceling headphones', 150.00, 50,'2024-07-15 21:19:17'),
('D521512', 'Smartwatch', 'Waterproof smartwatch', 200.00, 30,'2024-06-24 21:44:39'),
('G796234', 'Tablet', '10-inch screen tablet', 400.00, 20,'2025-03-07 22:00:10');
```

```
CREATE TABLE Orders (
    OrderID INT PRIMARY KEY,
    CustomerID INT,
    OrderDate TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    Status ENUM('Pending', 'Shipped', 'Delivered', 'Canceled') DEFAULT 'Pending',
    TotalAmount DECIMAL(10,2),
    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
);

INSERT INTO Orders (OrderID, CustomerID, Status, TotalAmount) VALUES
(5446, 001, 'Pending', 1200.00),
(3591, 002, 'Shipped', 800.00),
(7925, 003, 'Delivered', 150.00),
(3697, 004, 'Canceled', 200.00),
(8264, 005, 'Pending', 400.00);


CREATE TABLE OrderDetails (
     OrderID INT,
     ProductID VARCHAR(50),
     Quantity INT NOT NULL,
     Price DECIMAL(10,2) NOT NULL,
     PRIMARY KEY (OrderID, ProductID),
     FOREIGN KEY (OrderID) REFERENCES Orders(OrderID),
     FOREIGN KEY (ProductID) REFERENCES Products(ProductID)
);


INSERT INTO OrderDetails (OrderID, ProductID, Quantity, Price) VALUES
(5446, 'D521512', 1, 1200.00),
(3591, 'E102548', 1, 800.00),
(7925, 'H935477', 2, 150.00),
(3697, 'G796234' , 1, 200.00),
(8264, 'C368412' , 1, 400.00);

CREATE TABLE Payments (
    PaymentID INT PRIMARY KEY AUTO_INCREMENT,
    OrderID INT,
    PaymentDate TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    Amount DECIMAL(10,2) NOT NULL,
    PaymentMethod ENUM('Credit Card', 'Debit Card', 'PayPal', 'Bank Transfer') NOT NULL,
    FOREIGN KEY (OrderID) REFERENCES Orders(OrderID)
);

INSERT INTO Payments (OrderID, Amount, PaymentMethod) VALUES
```

```
(7925, 1200.00, 'Credit Card'),
(5446, 800.00, 'Debit Card'),
(8264, 150.00, 'PayPal'),
(3591, 200.00, 'Bank Transfer'),
(3697, 400.00, 'Credit Card');

CREATE TABLE SupportTickets (
    TicketID INT PRIMARY KEY AUTO_INCREMENT,
    CustomerID INT,
    Subject VARCHAR(255) NOT NULL,
    Description TEXT NOT NULL,
    Status ENUM('Open', 'In Progress', 'Closed') DEFAULT 'Open',
    CreatedAt TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
);

INSERT INTO SupportTickets (TicketID, CustomerID, Subject, Description, Status) VALUES
(1,004, 'Order Delay', 'My order is delayed, please update.', 'Open'),
(2,002, 'Payment Issue', 'Payment was deducted twice.', 'In Progress'),
(3,003, 'Product Defect', 'Received a defective product.', 'Open'),
(4,001, 'Cancellation Request', 'Want to cancel my order.', 'Closed'),
(5,005, 'Account Issue', 'Unable to login to my account.', 'Open');
```

2.4 Python Code

Paste the Python (Streamlit) code here:

```
import streamlit as st
import mysql.connector
import pandas as pd

conn = mysql.connector.connect(
    host="localhost",
    user="root",
    password="12345678",
    database="cdms"
)
cursor = conn.cursor()

st.set_page_config(page_title="CDMS", layout="wide")
st.title("📋 Customer Database Management System")

# Centering and fitting homepage image
col1, col2, col3 = st.columns([1, 4, 1])
```

```
with col2:

st.image("https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcS--GlCutMNTWYwqjnGtNlW-GlefaFgzKhRNQ&s", use_container_width=True)

st.header("WELCOME")
st.write("Efficient customer data handling is the backbone of any successful business.")
st.write("Customer Database Management System (CDMS) is a user-friendly, powerful, and secure tool designed to help you manage customer information seamlessly. ")
st.write("Built using SQL for database management, Python for backend processing, and Streamlit for an interactive web interface, this system simplifies customer data operations like storing, retrieving, updating, and analyzing—all in real-time.")

menu = st.sidebar.selectbox("Navigation", [
    "Search", "Customers", "Products", "Orders", "Payments", "Support Tickets"
])

def show_table(query, columns):
    cursor.execute(query)
    data = cursor.fetchall()
    df = pd.DataFrame(data, columns=columns)
    st.dataframe(df, use_container_width=True)

if menu == "Search":
    st.subheader("🔍 Search")
    search_type = st.selectbox("Search By", [
        "Customer ID", "First Name", "Last Name", "Phone Number",
        "Product ID", "Payment ID", "Ticket Number"
    ])
    search_term = st.text_input("Enter Search Term")

    if search_term:
        if search_type == "Customer ID":
            query = f"SELECT * FROM Customers WHERE CustomerID = '{search_term}'"
            show_table(query, ['CustomerID', 'FirstName', 'LastName', 'Email', 'Phone', 'Address', 'CreatedAt'])
        elif search_type == "First Name":
            query = f"SELECT * FROM Customers WHERE FirstName LIKE '%{search_term}%'"
            show_table(query, ['CustomerID', 'FirstName', 'LastName', 'Email', 'Phone', 'Address', 'CreatedAt'])
        elif search_type == "Last Name":
            query = f"SELECT * FROM Customers WHERE LastName LIKE '%{search_term}%'"
            show_table(query, ['CustomerID', 'FirstName', 'LastName', 'Email', 'Phone', 'Address', 'CreatedAt'])
        elif search_type == "Phone Number":
```

```
            query = f"SELECT * FROM Customers WHERE Phone LIKE '%{search_term}%'"
            show_table(query, ['CustomerID', 'FirstName', 'LastName', 'Email', 'Phone', 'Address',
'CreatedAt'])
        elif search_type == "Product ID":
            query = f"SELECT * FROM Products WHERE ProductID = '{search_term}'"
            show_table(query, ['ProductID', 'ProductName', 'Description', 'Price', 'Stock', 'CreatedAt'])
        elif search_type == "Payment ID":
            query = f"SELECT * FROM Payments WHERE PaymentID = '{search_term}'"
            show_table(query, ['PaymentID', 'OrderID', 'PaymentDate', 'Amount', 'PaymentMethod'])
        elif search_type == "Ticket Number":
            query = f"SELECT * FROM SupportTickets WHERE TicketID = '{search_term}'"
            show_table(query, ['TicketID', 'CustomerID', 'Subject', 'Description', 'Status', 'CreatedAt'])

if menu == "Customers":
    st.subheader("📇 Customer List")
    show_table("SELECT * FROM Customers", ['CustomerID', 'FirstName', 'LastName', 'Email', 'Phone',
'Address', 'CreatedAt'])

    st.markdown("---")
    st.subheader("➕ Add or Delete Customer")
    option = st.radio("Choose Action", ["Add Customer", "Delete Customer"])

    if option == "Add Customer":
        col1, col2 = st.columns(2)
        with col1:
            cust_id = st.number_input("Customer ID", step=1)
            first = st.text_input("First Name")
            last = st.text_input("Last Name")
        with col2:
            email = st.text_input("Email")
            phone = st.text_input("Phone")
            address = st.text_area("Address")

        if st.button("Add Customer"):
            try:
                cursor.execute("""
                    INSERT INTO Customers (CustomerID, FirstName, LastName, Email, Phone, Address)
                    VALUES (%s, %s, %s, %s, %s, %s)
                """, (cust_id, first, last, email, phone, address))
                conn.commit()
                st.success("Customer added successfully!")
            except Exception as e:
                st.error(f"Error: {e}")

    elif option == "Delete Customer":
```

```
        del_id = st.number_input("Enter Customer ID to delete", step=1)
        if st.button("Delete Customer"):
            try:
                cursor.execute("DELETE FROM Customers WHERE CustomerID = %s", (del_id,))
                conn.commit()
                st.success("Customer deleted successfully!")
            except Exception as e:
                st.error(f"Error: {e}")

if menu == "Products":
    st.subheader("📦 Product List")
    show_table("SELECT * FROM Products", ['ProductID', 'ProductName', 'Description', 'Price', 'Stock',
'CreatedAt'])

    st.markdown("---")
    st.subheader("➕ Add or Delete Product")
    option = st.radio("Choose Action", ["Add Product", "Delete Product"])

    if option == "Add Product":
        col1, col2 = st.columns(2)
        with col1:
            pid = st.text_input("Product ID")
            name = st.text_input("Product Name")
            desc = st.text_area("Description")
        with col2:
            price = st.number_input("Price", step=0.01)
            stock = st.number_input("Stock", step=1)

        if st.button("Add Product"):
            try:
                cursor.execute("""
                    INSERT INTO Products (ProductID, ProductName, Description, Price, Stock)
                    VALUES (%s, %s, %s, %s, %s)
                """, (pid, name, desc, price, stock))
                conn.commit()
                st.success("Product added successfully!")
            except Exception as e:
                st.error(f"Error: {e}")

    elif option == "Delete Product":
        del_pid = st.text_input("Enter Product ID to delete")
        if st.button("Delete Product"):
            try:
                cursor.execute("DELETE FROM Products WHERE ProductID = %s", (del_pid,))
                conn.commit()
```

```
            st.success("Product deleted successfully!")
        except Exception as e:
            st.error(f"Error: {e}")

if menu == "Orders":
    st.subheader("📦 Orders")
    show_table("SELECT * FROM Orders", ['OrderID', 'CustomerID', 'OrderDate', 'Status',
'TotalAmount'])

    st.markdown("---")
    st.subheader("➕ Add or Delete Order")
    option = st.radio("Choose Action", ["Add Order", "Delete Order"])

    if option == "Add Order":
        col1, col2 = st.columns(2)
        with col1:
            oid = st.number_input("Order ID", step=1)
            cid = st.number_input("Customer ID", step=1)
        with col2:
            status = st.selectbox("Status", ['Pending', 'Shipped', 'Delivered', 'Canceled'])
            total = st.number_input("Total Amount", step=0.01)

        if st.button("Add Order"):
            try:
                cursor.execute("""
                    INSERT INTO Orders (OrderID, CustomerID, Status, TotalAmount)
                    VALUES (%s, %s, %s, %s)
                """, (oid, cid, status, total))
                conn.commit()
                st.success("Order added successfully!")
            except Exception as e:
                st.error(f"Error: {e}")

    elif option == "Delete Order":
        del_oid = st.number_input("Enter Order ID to delete", step=1)
        if st.button("Delete Order"):
            try:
                cursor.execute("DELETE FROM Orders WHERE OrderID = %s", (del_oid,))
                conn.commit()
                st.success("Order deleted successfully!")
            except Exception as e:
                st.error(f"Error: {e}")

if menu == "Payments":
    st.subheader("💳 Payments")
```

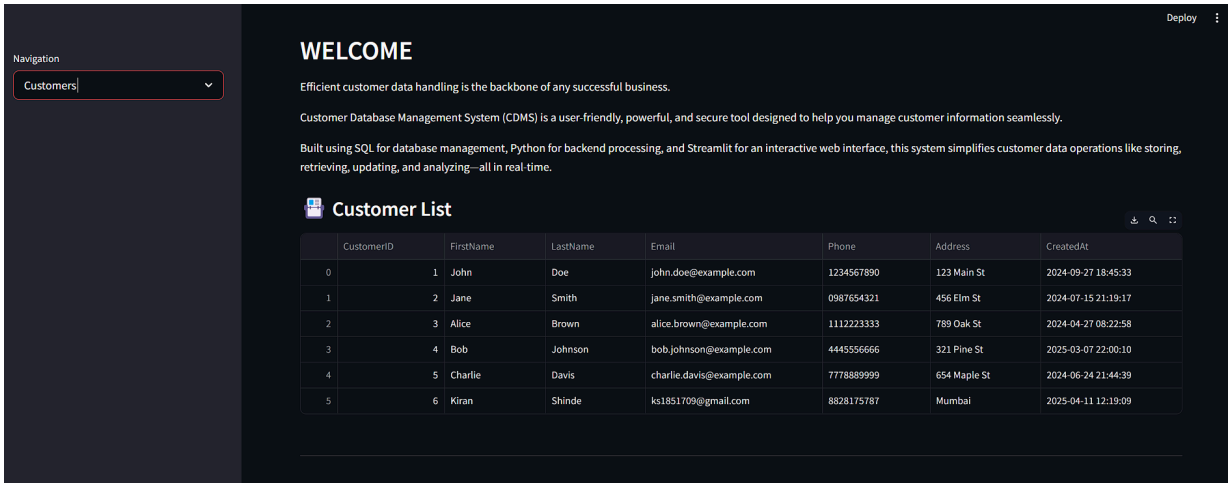
```
    show_table("SELECT * FROM Payments", ['PaymentID', 'OrderID', 'PaymentDate', 'Amount',
'PaymentMethod'])

    st.markdown("---")
    st.subheader("➕ Add or Delete Payment")
    option = st.radio("Choose Action", ["Add Payment", "Delete Payment"])

    if option == "Add Payment":
        oid = st.number_input("Order ID", step=1)
        amount = st.number_input("Amount", step=0.01)
        method = st.selectbox("Payment Method", ['Credit Card', 'Debit Card', 'PayPal', 'Bank Transfer'])

        if st.button("Add Payment"):
            try:
                cursor.execute("""
                    INSERT INTO Payments (OrderID, Amount, PaymentMethod)
                    VALUES (%s, %s, %s)
                """, (oid, amount, method))
                conn.commit()
                st.success("Payment recorded successfully!")
            except Exception as e:
                st.error(f"Error: {e}")

    elif option == "Delete Payment":
        del_pid = st.number_input("Enter Payment ID to delete", step=1)
        if st.button("Delete Payment"):
            try:
                cursor.execute("DELETE FROM Payments WHERE PaymentID = %s", (del_pid,))
                conn.commit()
                st.success("Payment deleted successfully!")
            except Exception as e:
                st.error(f"Error: {e}")

if menu == "Support Tickets":
    st.subheader("🎫 Support Tickets")
    show_table("SELECT * FROM SupportTickets", ['TicketID', 'CustomerID', 'Subject', 'Description',
'Status', 'CreatedAt'])

    st.markdown("---")
    st.subheader("➕ Add or Delete Ticket")
    option = st.radio("Choose Action", ["Add Ticket", "Delete Ticket"])

    if option == "Add Ticket":
        cust_id = st.number_input("Customer ID", step=1)
        subject = st.text_input("Subject")
```

```
    description = st.text_area("Description")
    status = st.selectbox("Status", ['Open', 'In Progress', 'Closed'])

    if st.button("Create Ticket"):
        try:
            cursor.execute("""
                INSERT INTO SupportTickets (CustomerID, Subject, Description, Status)
                VALUES (%s, %s, %s, %s)
            """, (cust_id, subject, description, status))
            conn.commit()
            st.success("Ticket submitted successfully!")
        except Exception as e:
            st.error(f"Error: {e}")

elif option == "Delete Ticket":
    del_tid = st.number_input("Enter Ticket ID to delete", step=1)
    reason = st.text_area("Reason for deletion")
    if st.button("Delete Ticket"):
        if reason:
            try:
                cursor.execute("DELETE FROM SupportTickets WHERE TicketID = %s", (del_tid,))
                conn.commit()
                st.success("Ticket deleted successfully!")
            except Exception as e:
                st.error(f"Error: {e}")
        else:
            st.warning("Please provide a reason for deletion.")
```

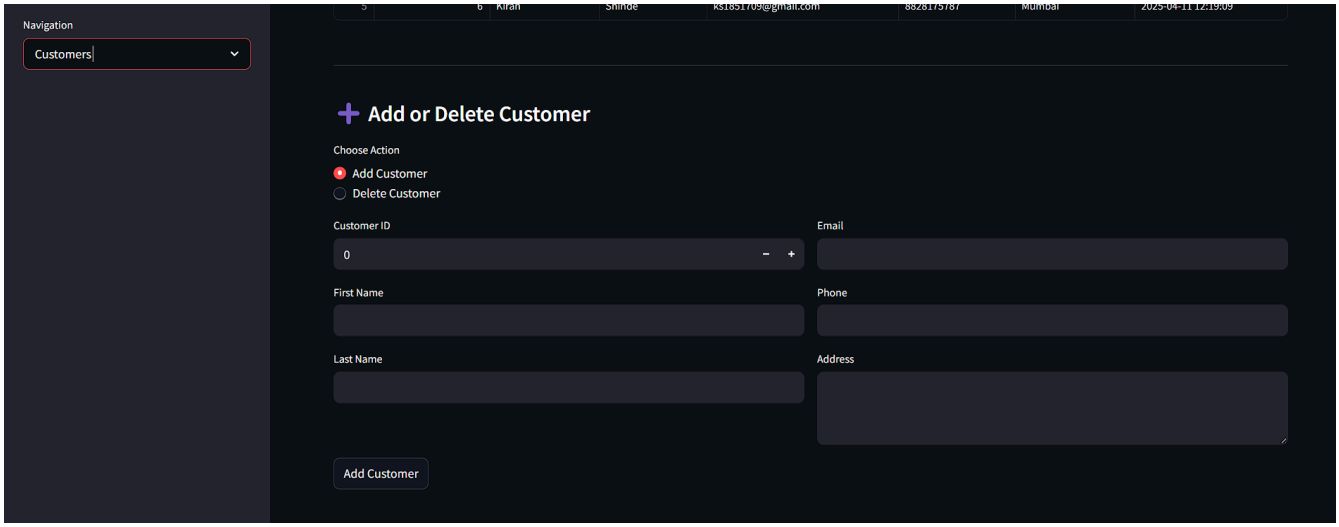
4. UI Screenshots

Paste the UI and output screenshots here:

1. Customer
    a. Here we have an option to add or delete customer as well along with showing customer list



2. Add/Delete Customer



Complete code for CDMS

```
import streamlit as st
import mysql.connector
import pandas as pd

conn = mysql.connector.connect(
    host="localhost",
    user="root",
    password="12345678",
    database="cdms"
)
cursor = conn.cursor()

st.set_page_config(page_title="CDMS", layout="wide")
st.title("📋 Customer Database Management System")

# Centering and fitting homepage image
col1, col2, col3 = st.columns([1, 4, 1])
with col2:

st.image("https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcS--GlCutMNTWYwqjnGtNlW-GlefaFgzKhRNQ&s", use_container_width=True)

st.header("WELCOME")
st.write("Efficient customer data handling is the backbone of any successful business.")
st.write("Customer Database Management System (CDMS) is a user-friendly, powerful, and secure tool designed to help you manage customer information seamlessly. ")
st.write("Built using SQL for database management, Python for backend processing, and Streamlit for an interactive web interface, this system simplifies customer data operations like storing, retrieving, updating, and analyzing—all in real-time.")

menu = st.sidebar.selectbox("Navigation", [
    "Search", "Customers", "Products", "Orders", "Payments", "Support Tickets"
])

def show_table(query, columns):
    cursor.execute(query)
    data = cursor.fetchall()
    df = pd.DataFrame(data, columns=columns)
    st.dataframe(df, use_container_width=True)

if menu == "Search":
```

```
    st.subheader("🔍 Search")
    search_type = st.selectbox("Search By", [
        "Customer ID", "First Name", "Last Name", "Phone Number",
        "Product ID", "Payment ID", "Ticket Number"
    ])
    search_term = st.text_input("Enter Search Term")

    if search_term:
        if search_type == "Customer ID":
            query = f"SELECT * FROM Customers WHERE CustomerID = '{search_term}'"
            show_table(query, ['CustomerID', 'FirstName', 'LastName', 'Email', 'Phone', 'Address',
'CreatedAt'])
        elif search_type == "First Name":
            query = f"SELECT * FROM Customers WHERE FirstName LIKE '%{search_term}%'"
            show_table(query, ['CustomerID', 'FirstName', 'LastName', 'Email', 'Phone', 'Address',
'CreatedAt'])
        elif search_type == "Last Name":
            query = f"SELECT * FROM Customers WHERE LastName LIKE '%{search_term}%'"
            show_table(query, ['CustomerID', 'FirstName', 'LastName', 'Email', 'Phone', 'Address',
'CreatedAt'])
        elif search_type == "Phone Number":
            query = f"SELECT * FROM Customers WHERE Phone LIKE '%{search_term}%'"
            show_table(query, ['CustomerID', 'FirstName', 'LastName', 'Email', 'Phone', 'Address',
'CreatedAt'])
        elif search_type == "Product ID":
            query = f"SELECT * FROM Products WHERE ProductID = '{search_term}'"
            show_table(query, ['ProductID', 'ProductName', 'Description', 'Price', 'Stock', 'CreatedAt'])
        elif search_type == "Payment ID":
            query = f"SELECT * FROM Payments WHERE PaymentID = '{search_term}'"
            show_table(query, ['PaymentID', 'OrderID', 'PaymentDate', 'Amount', 'PaymentMethod'])
        elif search_type == "Ticket Number":
            query = f"SELECT * FROM SupportTickets WHERE TicketID = '{search_term}'"
            show_table(query, ['TicketID', 'CustomerID', 'Subject', 'Description', 'Status', 'CreatedAt'])

if menu == "Customers":
    st.subheader("📇 Customer List")
    show_table("SELECT * FROM Customers", ['CustomerID', 'FirstName', 'LastName', 'Email', 'Phone',
'Address', 'CreatedAt'])

    st.markdown("---")
    st.subheader("➕ Add or Delete Customer")
    option = st.radio("Choose Action", ["Add Customer", "Delete Customer"])

    if option == "Add Customer":
        col1, col2 = st.columns(2)
```

```
        with col1:
            cust_id = st.number_input("Customer ID", step=1)
            first = st.text_input("First Name")
            last = st.text_input("Last Name")
        with col2:
            email = st.text_input("Email")
            phone = st.text_input("Phone")
            address = st.text_area("Address")

        if st.button("Add Customer"):
            try:
                cursor.execute("""
                    INSERT INTO Customers (CustomerID, FirstName, LastName, Email, Phone, Address)
                    VALUES (%s, %s, %s, %s, %s, %s)
                """, (cust_id, first, last, email, phone, address))
                conn.commit()
                st.success("Customer added successfully!")
            except Exception as e:
                st.error(f"Error: {e}")

    elif option == "Delete Customer":
        del_id = st.number_input("Enter Customer ID to delete", step=1)
        if st.button("Delete Customer"):
            try:
                cursor.execute("DELETE FROM Customers WHERE CustomerID = %s", (del_id,))
                conn.commit()
                st.success("Customer deleted successfully!")
            except Exception as e:
                st.error(f"Error: {e}")

if menu == "Products":
    st.subheader("📦 Product List")
    show_table("SELECT * FROM Products", ['ProductID', 'ProductName', 'Description', 'Price', 'Stock',
'CreatedAt'])

    st.markdown("---")
    st.subheader("➕ Add or Delete Product")
    option = st.radio("Choose Action", ["Add Product", "Delete Product"])

    if option == "Add Product":
        col1, col2 = st.columns(2)
        with col1:
            pid = st.text_input("Product ID")
            name = st.text_input("Product Name")
            desc = st.text_area("Description")
```

```
        with col2:
            price = st.number_input("Price", step=0.01)
            stock = st.number_input("Stock", step=1)

        if st.button("Add Product"):
            try:
                cursor.execute("""
                    INSERT INTO Products (ProductID, ProductName, Description, Price, Stock)
                    VALUES (%s, %s, %s, %s, %s)
                """, (pid, name, desc, price, stock))
                conn.commit()
                st.success("Product added successfully!")
            except Exception as e:
                st.error(f"Error: {e}")

    elif option == "Delete Product":
        del_pid = st.text_input("Enter Product ID to delete")
        if st.button("Delete Product"):
            try:
                cursor.execute("DELETE FROM Products WHERE ProductID = %s", (del_pid,))
                conn.commit()
                st.success("Product deleted successfully!")
            except Exception as e:
                st.error(f"Error: {e}")

if menu == "Orders":
    st.subheader("📦 Orders")
    show_table("SELECT * FROM Orders", ['OrderID', 'CustomerID', 'OrderDate', 'Status',
'TotalAmount'])

    st.markdown("---")
    st.subheader("➕ Add or Delete Order")
    option = st.radio("Choose Action", ["Add Order", "Delete Order"])

    if option == "Add Order":
        col1, col2 = st.columns(2)
        with col1:
            oid = st.number_input("Order ID", step=1)
            cid = st.number_input("Customer ID", step=1)
        with col2:
            status = st.selectbox("Status", ['Pending', 'Shipped', 'Delivered', 'Canceled'])
            total = st.number_input("Total Amount", step=0.01)

        if st.button("Add Order"):
            try:
```

```
            cursor.execute("""
                INSERT INTO Orders (OrderID, CustomerID, Status, TotalAmount)
                VALUES (%s, %s, %s, %s)
            """, (oid, cid, status, total))
            conn.commit()
            st.success("Order added successfully!")
        except Exception as e:
            st.error(f"Error: {e}")

    elif option == "Delete Order":
        del_oid = st.number_input("Enter Order ID to delete", step=1)
        if st.button("Delete Order"):
            try:
                cursor.execute("DELETE FROM Orders WHERE OrderID = %s", (del_oid,))
                conn.commit()
                st.success("Order deleted successfully!")
            except Exception as e:
                st.error(f"Error: {e}")

if menu == "Payments":
    st.subheader("💳 Payments")
    show_table("SELECT * FROM Payments", ['PaymentID', 'OrderID', 'PaymentDate', 'Amount',
'PaymentMethod'])

    st.markdown("---")
    st.subheader("➕ Add or Delete Payment")
    option = st.radio("Choose Action", ["Add Payment", "Delete Payment"])

    if option == "Add Payment":
        oid = st.number_input("Order ID", step=1)
        amount = st.number_input("Amount", step=0.01)
        method = st.selectbox("Payment Method", ['Credit Card', 'Debit Card', 'PayPal', 'Bank Transfer'])

        if st.button("Add Payment"):
            try:
                cursor.execute("""
                    INSERT INTO Payments (OrderID, Amount, PaymentMethod)
                    VALUES (%s, %s, %s)
                """, (oid, amount, method))
                conn.commit()
                st.success("Payment recorded successfully!")
            except Exception as e:
                st.error(f"Error: {e}")

    elif option == "Delete Payment":
```

```
        del_pid = st.number_input("Enter Payment ID to delete", step=1)
        if st.button("Delete Payment"):
            try:
                cursor.execute("DELETE FROM Payments WHERE PaymentID = %s", (del_pid,))
                conn.commit()
                st.success("Payment deleted successfully!")
            except Exception as e:
                st.error(f"Error: {e}")

if menu == "Support Tickets":
    st.subheader("🎫 Support Tickets")
    show_table("SELECT * FROM SupportTickets", ['TicketID', 'CustomerID', 'Subject', 'Description',
'Status', 'CreatedAt'])

    st.markdown("---")
    st.subheader("➕ Add or Delete Ticket")
    option = st.radio("Choose Action", ["Add Ticket", "Delete Ticket"])

    if option == "Add Ticket":
        cust_id = st.number_input("Customer ID", step=1)
        subject = st.text_input("Subject")
        description = st.text_area("Description")
        status = st.selectbox("Status", ['Open', 'In Progress', 'Closed'])

        if st.button("Create Ticket"):
            try:
                cursor.execute("""
                    INSERT INTO SupportTickets (CustomerID, Subject, Description, Status)
                    VALUES (%s, %s, %s, %s)
                """, (cust_id, subject, description, status))
                conn.commit()
                st.success("Ticket submitted successfully!")
            except Exception as e:
                st.error(f"Error: {e}")

    elif option == "Delete Ticket":
        del_tid = st.number_input("Enter Ticket ID to delete", step=1)
        reason = st.text_area("Reason for deletion")
        if st.button("Delete Ticket"):
            if reason:
                try:
                    cursor.execute("DELETE FROM SupportTickets WHERE TicketID = %s", (del_tid,))
                    conn.commit()
                    st.success("Ticket deleted successfully!")
                except Exception as e:
```

```
            st.error(f"Error: {e}")
    else:
        st.warning("Please provide a reason for deletion.")
```